Named Entity Recognition for Question Answering - CS 378 Fall 2020

Avinash Damania and Abhishek Khare
{avinash.damania, abhishek.khare}@utexas.edu

Abstract

Question Answering is a rapidly evolving field, with a lot of major advancements coming out in the last five years. The existing QA system adapted from the DrQA paper (Chen et al., 2017) has moderate success, however we believed it was lacking important contexts that could boost performance. In this paper, we propose utilizing name entity recognition (NER) to contextualize questions and answers to better match correct spans to questions. We trained and tested these contextualizations on the SQuAD adversarial dataset, and found some minimal improvements in performance. Our findings detail our two approaches and their performances individually and combined through a set of experiments.

1 Introduction

Given a question answering system, our idea for improving its performance was integrating Named Entity Recognition (NER) in order to minimize wrong answers and help find answers that overlap more with the given questions. This approach could especially help in adversarial datasets like the given SQuAD dataset. Our initial plan was to run NER on a group of the top k potential answers (the different spans from different start/end distributions), and compare this to the result of running NER on the question. We hypothesized that answers that have more named entities in common with the question are more likely to be correct. In addition, for questions with "who" and "where" interrogatives, we are planning to prioritize choosing answers if they have a named entity of a category that corresponds to the question's interrogative, which we believe will help the performance of the system as well.

Apart from simply integrating NER into the existing question answering system, we must also consider how to use it to improve the existing probability distribution analysis that the system uses to choose an answer. In other words, we don't want to entirely replace the existing evaluation of potential answers - we want to still take into account the highest probability answers and use NER as a way to prioritize between already close best answers. Therefore, we will develop a form of a voting system/compounded probability multipliers that combines the information from the base QA system and the results of running NER on the question/answers to choose a best overall answer.

2 Research/Prior Work

Prior research has shown NER to be successful in boosting different aspects of QA performance. NER has helped preprocess data to reduce QA input (Noguera et al., 2005), which helps with the efficiency of the QA system. In addition, as briefly noted by (Mendes et al., 2010), NER can be used to map question types to entity labels, meaning that certain questions should have an answer that falls into some specific subset of named entity categories. This idea became the basis for the second approach that we implemented and tested.

We took inspiration from these implementations, and decided to use NER to directly improve a system's EM/F1 performance in adversarial settings rather than improving the efficiency of the system. As made apparent by the research we found, NER can definitely help the performance of the baseline QA system through counting common named entities and mapping question types to categories of named entities.

3 Implementation

The integration of Named Entity Recognition into the base QA system required no changes to the model. Most of the changes were in the write_predictions method, which calculates answers to questions based on the precomputed

probabilities from the QA system. We implemented a topk_span_endpoints function that returns a list of k optimal answers instead of returning just one. To test this, we changed the call to search_span_endpoints in write_predictions in main.py to instead use topk_span_endpoints[0], which logically should give the best possible answer and lead to the same EM/F1 scores when using the given small baseline model. We found that the scores were the same ('EM': 47.21, 'F1': 59.81), so our function was working as expected.

We also wrote code to perform NER on the potential answers and compare them to the question text in write_predictions as well. This code takes a start/end index and finds the sentence(s) surrounding it so we can search for named entities in the area around a potential answer span and not miss out on the context around an answer that could help the system.

One issue we ran into was that spaCy operates best on truecased text, and the write_predictions method is given lowercased text. To fix this, we modified the QADataset class in data.py to take a lowercase boolean argument. This allowed us to maintain both lowercased and truecased datasets for usage in main.py.

The bulk of the implementation details and changes are in the main.py file, which uses a trained model and training dataset to make predictions for the answers to given questions in a development dataset. In write_predictions, we first make a call to our topk answers function to get a list of the best answers instead of just one, as the baseline system did. We also thread in the truecased dataset and get the question text from the dataset. We then pass this topk list through our two approaches/ideas for applying NER to improve the system.

The first approach involved counting the number of named entities an answer span has in common with the question. This is achieved by using a max heap, which then gives us the answers sorted by the number of common entities. We then apply a multiplier to the probabilities that the base system calculates in order to weight answers higher if they have more entities in common with the question. In order to do this, we created a simple calculate_multiplier_increments function in utils.py that takes a range from a lower value to an upper value and creates evenly incremented multipliers to apply to a list of k probabilities. For example, for a lower of 1.0 and an upper of 1.5, we will get a list of multipliers that looks like [1.5, 1.4, 1.3, 1.2, 1.1].

The second approach involved finding the first interrogative in the question (ex: "who", "what") and finding named entities in the answer that are of the proper category to answer the question. We created a map from interrogatives to their corresponding named entity categories in spaCy. For example, "who" maps to the categories "PERSON", "NORP" (nationalities or religious or political groups), and "ORG" (companies, agencies, institutions, etc.), which means that named entities in those categories are more likely to be part of an answer to a "who" question. We apply a multiplier in a similar fashion to each answer once they've been reranked by this second approach, and return the answers with their new probabilities.

We modularized the functionality for the two approaches into separate methods that operate with list inputs and list outputs. This allowed us to easily evaluate using one approach or the other on its own, as well as testing both approaches together. After implementing the code for this project, we turned to finding metrics for evaluation, and thought about what would be best to tune and test over.

4 Evaluation/Metrics

For evaluating our system, we chose to use the EM (exact match) and F1 score as metrics. The baseline QA system comes with code that calculates these metrics and gives us a good standard to compare our modified system's performance to. One thing that helped us a lot during development to figure out how exactly our system was working was to thread the gold standard answers from the testing dataset to our write_predictions method. When our system returned an answer different from the baseline, we were able to compare this to the gold answer and see the question/context that our system performed better in, as well as what the correct answers were supposed to be. We used this approach to find specific examples that show the differences between our system versus the baseline system; we talk more in depth about two of these examples in the Analysis section later in this paper.

As for testing and gathering data/results, we wanted our system to perform better on the adversarial dataset questions, so we tested our system on both the SQuAD (Rajpurkar et al., 2016) and adversarial SQuAD (Jia and Liang, 2017) datasets provided to us. We evaluated the four combinations of our two ideas for NER integration: the baseline

system with neither idea, the first idea alone, the second idea alone, and then both ideas together. We also tuned the upper bound for the multipliers, trying the values 1.1, 1.3, 1.5, and 5.0. More extreme values proved to only perform worse, so we limited the range to the values listed above. During testing, we only used the small pretrained model provided (Durrett et al., 2020), since our changes to the system weren't concerned with the model itself. Instead, we were testing how our changes performed in comparison to a baseline system, so as long as the baseline is constant, we decided it was fine to use the small model for faster runtimes. In order to evaluate our system efficiently, we used Google Colab to run the main.py command on the two datasets.

5 Results/Data

On the right side of this page, the results are displayed in six separate tables for ease of viewing. The first three tables contain data from testing on the SQuAD dataset, while the last three tables are from testing on the adversarial SQuAD dataset. For each set of three tables, we tested using both approaches, the first approach alone, and the second approach alone. The baseline system's performance on each of the two overall datasets is also included at the top of each table for ease of comparison to our system's performance.

Overall, the results show not too drastic of a difference in performance between our system and the baseline system. What we were testing was how good each of the approaches were individually and together compared to the baseline, as well as what multiplier we should use (essentially how much to weight the named entity recognition in choosing the best answer for a given question). We also wanted to evaluate how much our system improved upon the baseline system for the adversarial dataset, as that was where the baseline system struggled more. We found that an upper bound of 1.1 on the multiplier generally performed the best across both datasets. We discuss the results of our system and look at some specific examples of questions where our system performed better and worse in the next section.

6 Analysis

The experiments made immediately clear that the combination of both approaches resulted in worse performance than our two approaches individually.

Multiplier upper bound	EM/F1 scores
baseline	48.73, 61.00
1.1	48.7, 60.98
1.3	48.64, 60.95
1.5	48.69, 60.96
5.0	48.10, 60.74

Table 1: SQuAD dataset, both approaches.

Multiplier upper bound	EM/F1 scores
baseline	48.73, 61.00
1.1	48.72, 61.00
1.3	48.66, 60.96
1.5	48.74, 61.00
5.0	48.42, 60.89

Table 2: SQuAD dataset, first approach only.

Multiplier upper bound	EM/F1 scores
baseline	48.73, 61.00
1.1	48.73, 61.00
1.3	48.72, 60.98
1.5	48.69, 60.96
5.0	48.62, 60.93

Table 3: SQuAD dataset, second approach only.

Multiplier upper bound	EM/F1 scores
baseline	36.71, 46.53
1.1	36.65, 46.49
1.3	36.49, 46.39
1.5	36.49, 46.36
5.0	36.09, 46.54

Table 4: Adversarial SQuAD dataset, both approaches.

Multiplier upper bound	EM/F1 scores
baseline	36.71, 46.53
1.1	36.65, 46.49
1.3	36.60, 46.44
1.5	36.60, 46.43
5.0	36.21, 46.47

Table 5: Adversarial SQuAD dataset, first approach only.

Multiplier upper bound	EM/F1 scores
baseline	36.71, 46.53
1.1	36.71, 46.53
1.3	36.71, 46.53
1.5	36.71, 46.53
5.0	36.60, 46.51

Table 6: Adversarial SQuAD dataset, second approach only.

For all but one of the experiments, the approaches individually underperformed the baseline, so likely their negative deviations compounded to make the combination even worse.

Out of the two approaches, it appears the common entities (first) approach produced results furthest deviating from the baseline performance. While mostly underperforming the second approach, it also produced our only system that outperformed the baseline. The best performing system involved the first approach only with a 1.5 weight on the SQuAD corpus. This system improved the EM by 0.1 with no reduction in F1 score. The success of this approach was dependent on the baseline system selecting spans from the incorrect sentence, and ours correcting for this. However, in most cases with an incorrect baseline answer, the answer was already derived from the correct sentence; it was just the wrong span. In hindsight, common entity matching would have likely been better as a preprocessing approach to reduce the number of spans to test, rather than a rescoring metric.

The interrogative matching (second) approach had more conservative results that didn't significantly deviate from the baseline. This one had some success in prioritizing the correct span out of multiple ones from the same sentence. As we found in specific examples, however, it also had a bias for longer spans that ultimately led to performance below the baseline in all experiments. This was a result of maximizing the number of entity labels mapping to the question interrogative, so a longer span with more entities would have a higher chance of matching labels even if the extraneous entities had no affiliation to the question. A future implementation of this approach would have to include adding weight to shorter spans to cancel out this effect, or weighting named entities based on proximity to the answer span.

We now discuss two specific examples of our system having a different answer than the baseline system.

6.1 Example 1: NER Integration Found Better Answer

CONTEXT: "...Richard Harbison 's purely morphological analysis in 1985 concluded that the cydippids are not monophyletic , in other words do not contain all and only the descendants of a single common ancestor that was itself a cydippid. ..."

QUESTION: Who did a morphologically analysis in 1985 that concluded cydippids are not monophyletic?

BASELINE ANSWER: harbison 's purely morphological analysis in 1985 concluded that the cydippids

OUR ANSWER: richard harbison

GOLD: ['richard harbison']

In this example, the baseline system answer only included the last name causing NER to reclassify 'harbison' as a product (PRODUCT entity category). The NER system in our system correctly classified the 'richard harbison' span as a name (PERSON entity category) and weighted it higher, resulting in it having the new highest probability. To reiterate from earlier, our system maps "who" questions to the entity categories of "PERSON", "NORP", and "ORG".

6.2 Example 2: NER Integration Got Worse Answer

CONTEXT: "...This was organised by the Council of Industrial Design established by the British government in 1944 " to promote by all practicable means the improvement of design in the products of British industry. ..."

QUESTION: Who organized the Britain Can Make It exhibition?

BASELINE ANSWER: council of industrial design

OUR ANSWER: council of industrial design established by the british government

GOLD: ['festival of britain (1951)', 'council of industrial design', 'the council of industrial design']

In this example, the interrogative matcher prioritized the most occurrences of the mapped entity labels. The longer baseline span containing an irrelevant entity, 'the british government', was weighted higher in our system as a result of it having an additional valid label for a "who" question. The phrase "council of industrial design" is tagged as "ORG", while "british" is tagged as "NORP". Our answer span has two entities of the correct category for a "who" question, meaning that it was ranked higher than the baseline system's answer that only had one correctly categorized entity.

7 Conclusions and Future Work

In the future, NER can continue to be integrated into QA systems to improve their performance on adversarial datasets. After analyzing our results, we had ideas for future improvements to the implementation of our approaches. The second approach could be improved by calculating the percentage of each answer span that consists of common named entities rather than counting the total number of common entities. This serves to penalize long answers, which our system gives preference to at the moment. Another solution to this problem would be weighting a named entity's impact on the span's probabilities based on the entity's proximity to the answer span itself. Common entity matching could also be moved to a preprocessing step to improve the efficiency of the system and reduce the number of potential answer spans to test, as mentioned earlier.

Our work in integrating NER into the baseline QA system did, to an extent, prove our hypothesis that NER can improve question answering systems in adversarial settings. Even though our our system mostly under-performed the baseline EM/F1 scores, specific examples of our system show that NER can sometimes find a better answer than the baseline QA system provided.

8 Acknowledgments

There are some credits we would like to attribute before listing our references. The baseline question answering system for this project was provided by Shrey Desai, Yasumasa Onoe, and Greg Durrett at https://github.com/gregdurrett/cs378nlp-sp20-fp. For named entity recognition, the spaCy library at https://spacy.io/ was used in our work. Google Colab was used to speed up training times. The template for this paper was created by (Bethard et al., 2020) for the 2020 ACL Conference.

We also wanted to say thank you to the professor (Greg Durrett) and other instructors (Tanya Goyal, Shivang Singh) who made CS 378 a possibility this fall, despite the challenges of moving an entire course to an online format. Your instruction and help throughout the semester is much appreciated!

References

Steven Bethard, Ryan Cotterrell, and Rui Yan. 2020. Acl 2020 proceedings template. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ana Cristina Mendes, Luísa Coheur, and Paula Vaz Lobo. 2010. Named entity recognition in questions: Towards a golden collection. In *Proceedings of the International Conference on Language Resources and Evaluation*.

Elisa Noguera, Antonio Toral, Fernando Llopis, and Rafael Muñoz. 2005. Reducing question answering input data using named entity recognition. In *Proceedings of the 8th International Conference on Text, Speech Dialogue*, pages 428–434.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.